CSCI 262
Data Structures

7 – Queues

“First in, first out”

Queues are a FIFO (first in, first out) structure. Think of a line of people waiting their turn:

If people are polite, the first in line is done first.

Queue vs. Stack

Stack. All interactions are with the top of the stack.

Queue: items are added to the back (or rear) and taken from the front.

Operations

• Adding an item to a queue: enqueue
• Removing an item from a queue: de-queue

A Simple Queue Class

```cpp
class queue {
public:
    char front();
    void enqueue(char c);
    void dequeue();
    size_t size();
    bool empty(); // true if queue is empty

private:
    // private stuff
};
```

The operations enqueue and dequeue are sometimes called push and pop. Don’t confuse a queue with a stack, though!
Using Queues

What does this code do?

```cpp
queue letters;
string text = "Data structures";
for (int j = 0; j < text.length(); j++) {
    letters.enqueue(text[j]);
}
while (!letters.empty()) {
    cout << letters.front();
    letters.dequeue();
}
```

Uses for Queues

Anywhere you need to keep things in order, particularly by time of arrival:
- Buffering character input
- Print jobs
- Process scheduling
- I/O request scheduling
- Web page request servicing
- Event handling (GUI, simulations, etc.)

StanfordCPPLib Queue

```cpp
#include "queue.h"

class Queue<ValueType>
{
    Operations:
    enqueue(ValueType v) // add value to end
    dequeue() // return and remove front value
    peek() // return front value
    size() // return number of elements
    isEmpty() // true if no elements
    clear() // remove all elements
    toString() // return a string representation
};
```